Prior Art

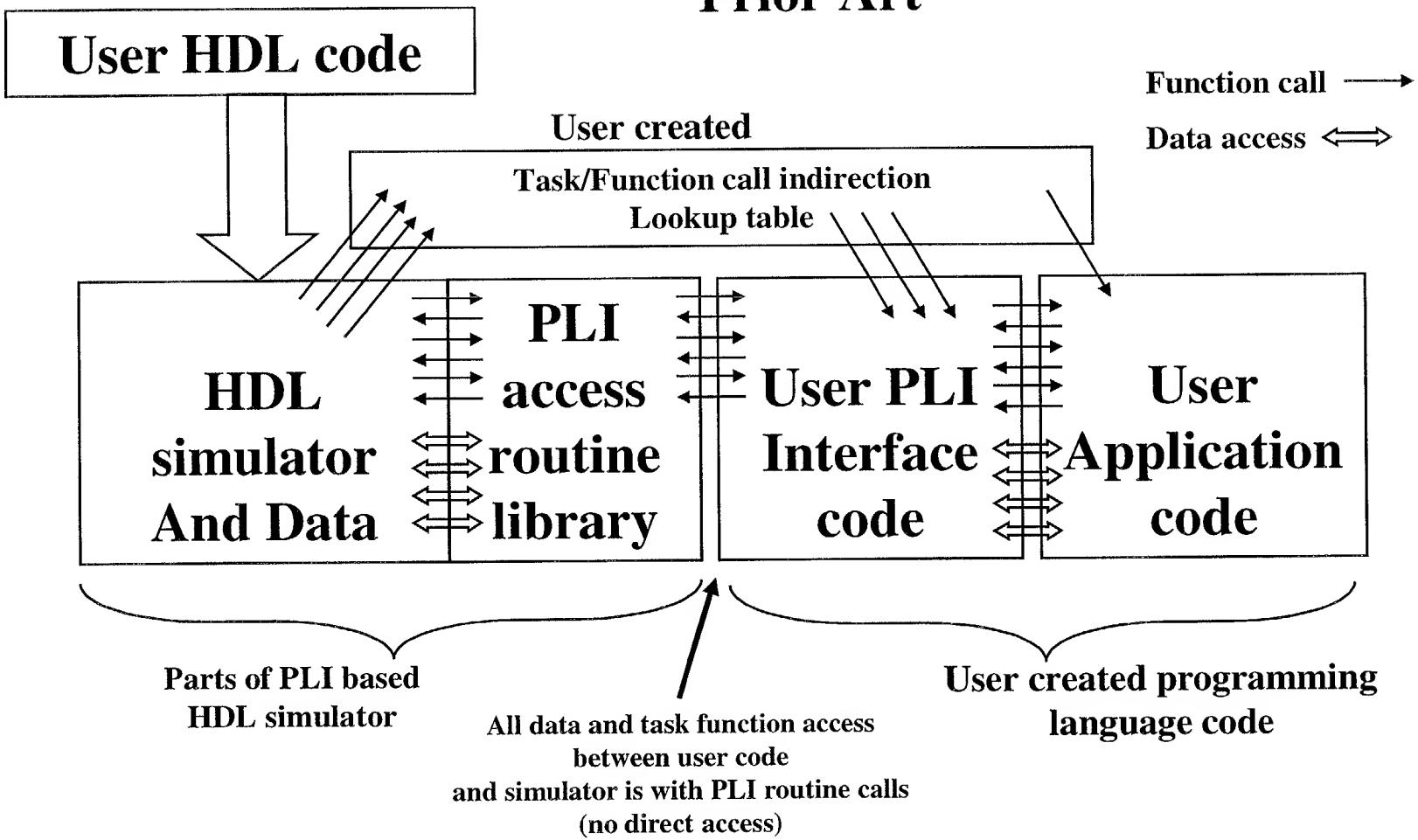


FIG. 1

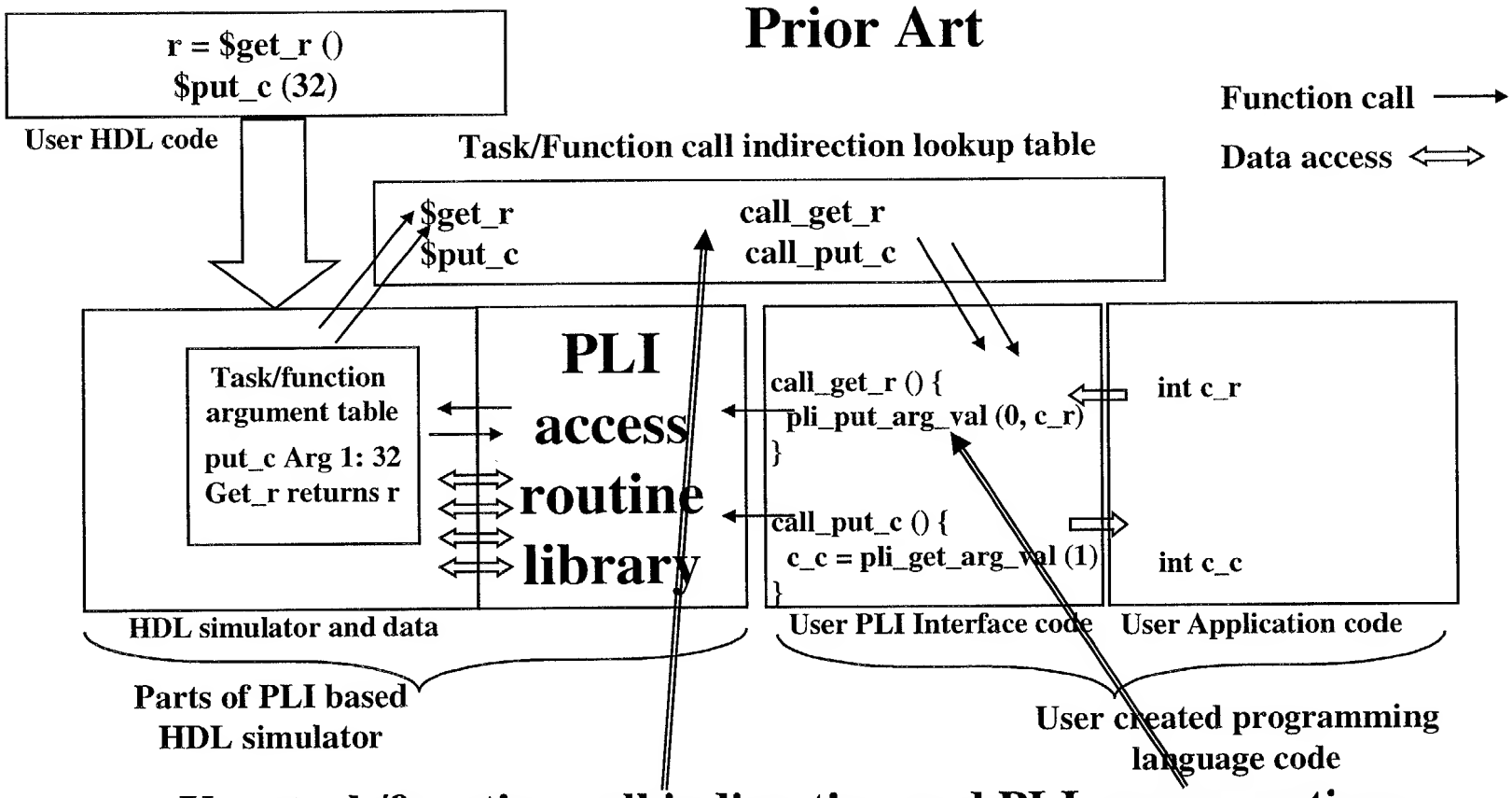

Prior Art
r = $get_r ()
$put_c (32)
User HDL code
Task/Function call indirection lookup table
$get_r
$put_c
call_get_r
call_put_c
Function call
Data access
Task/function argument table
put_c Arg 1: 32
Get_r returns r
PLI access routine library
call_get_r () {
pli_put_arg_val (0, c_r)
}
call_put_c () {
c_c = pli_get_arg_val (1)
}
int c_r
int c_c
HDL simulator and data
User PLI Interface code
User Application code
Parts of PLI based HDL simulator
User created programming language code
Uses task/function call indirection and PLI access routines to obtain & return variable values (no direct access)

FIG. 2

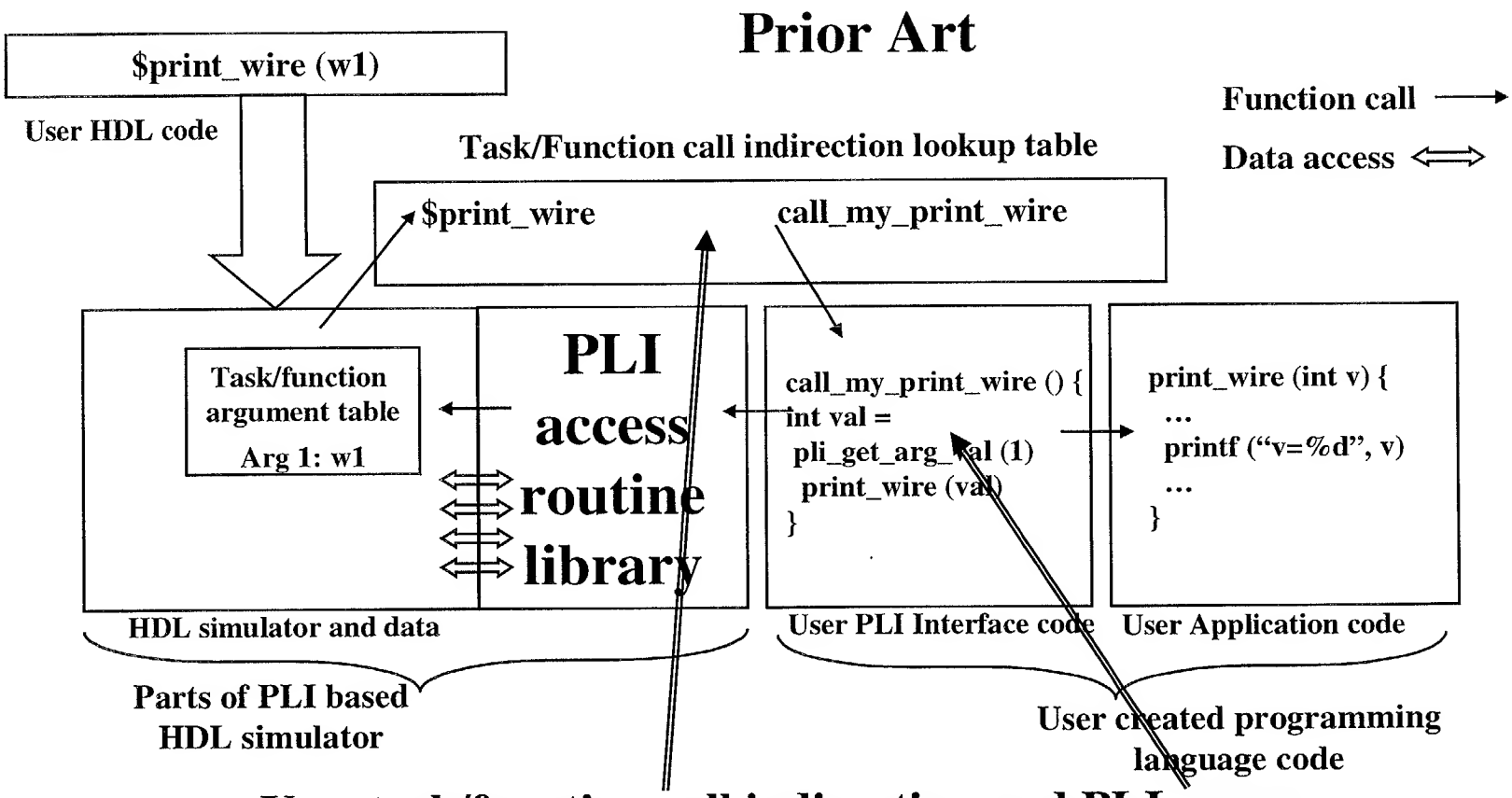

Prior Art
$print_wire (w1)
User HDL code
Function call
Data access
Task/Function call indirection lookup table
$print_wire
call_my_print_wire
Task/function argument table
Arg 1: w1
PLI access routine library
call_my_print_wire () {
int val =
pli_get_arg_val (1)
print_wire (val)
}
print_wire (int v) {
...
printf ("v=%d", v)
...
}
HDL simulator and data
User PLI Interface code
User Application code
Parts of PLI based HDL simulator
User created programming language code
Uses task/function call indirection and PLI access routine to obtain variable value (no direct access)

FIG. 3

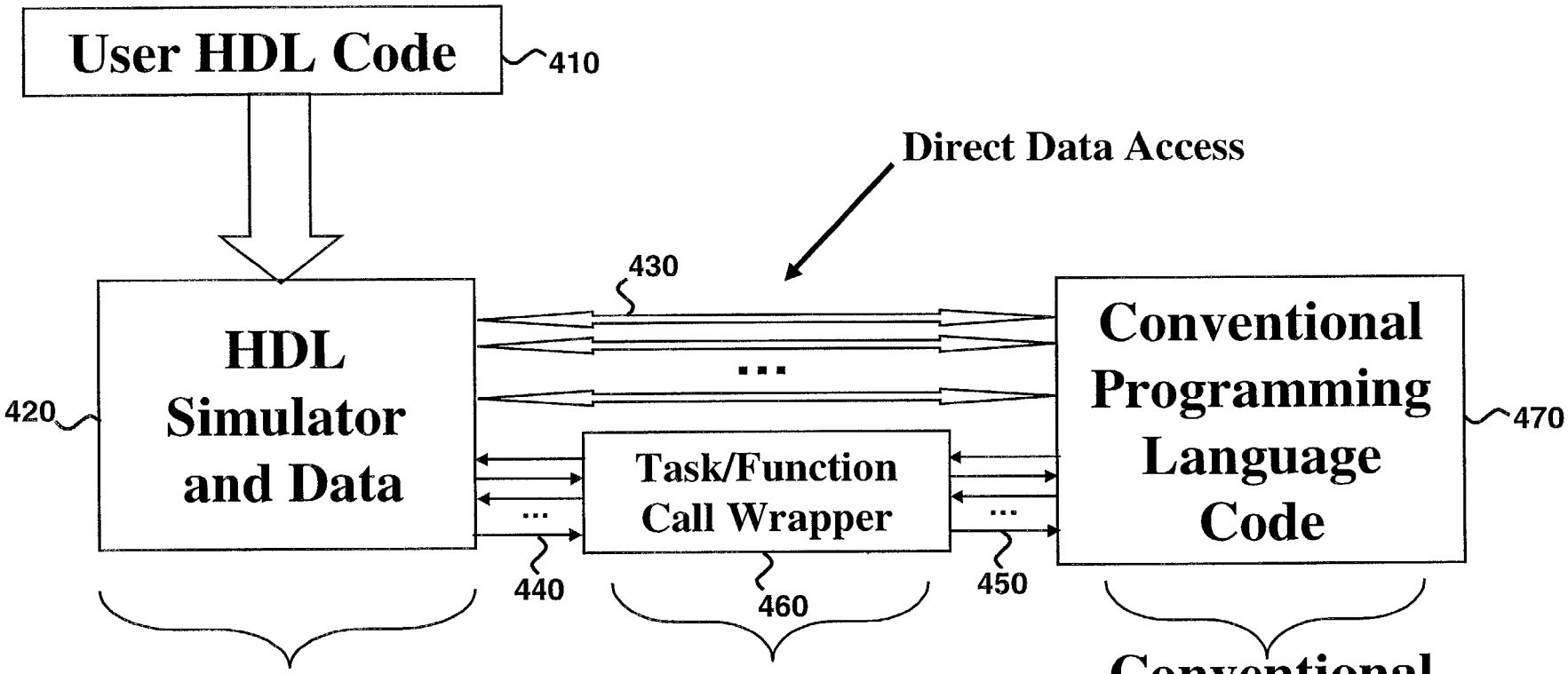

User HDL Code
410
Direct Data Access
430
HDL Simulator and Data
420
Task/Function Call Wrapper
Conventional Programming Language Code
470
440
460
450
Simulator
Automatically Generated
Conventional Programming Code, Including User Code

FIG. 4

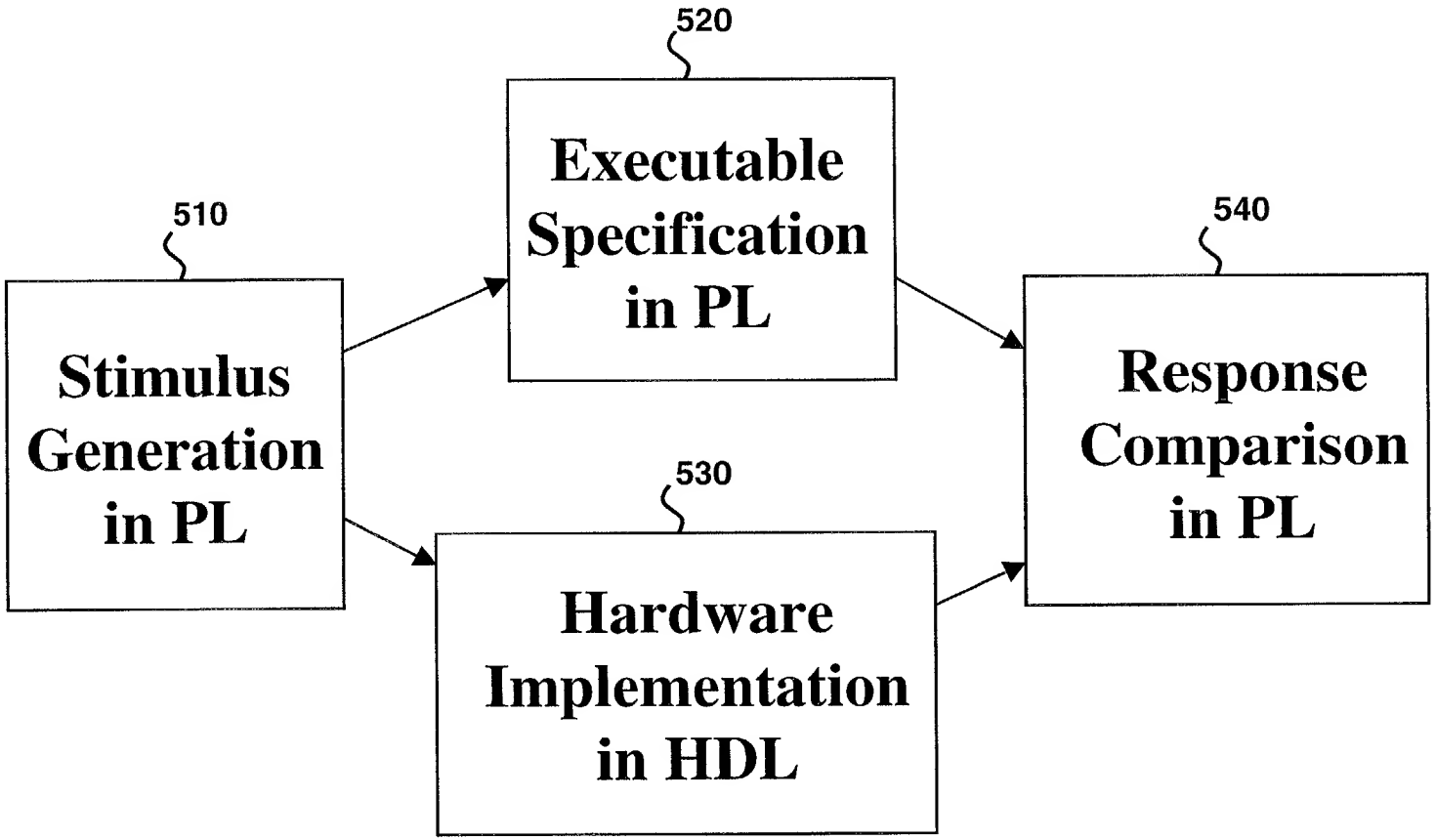

520
Executable Specification in PL
510
Stimulus Generation in PL
540
Response Comparison in PL
530
Hardware Implementation in HDL

FIG. 5

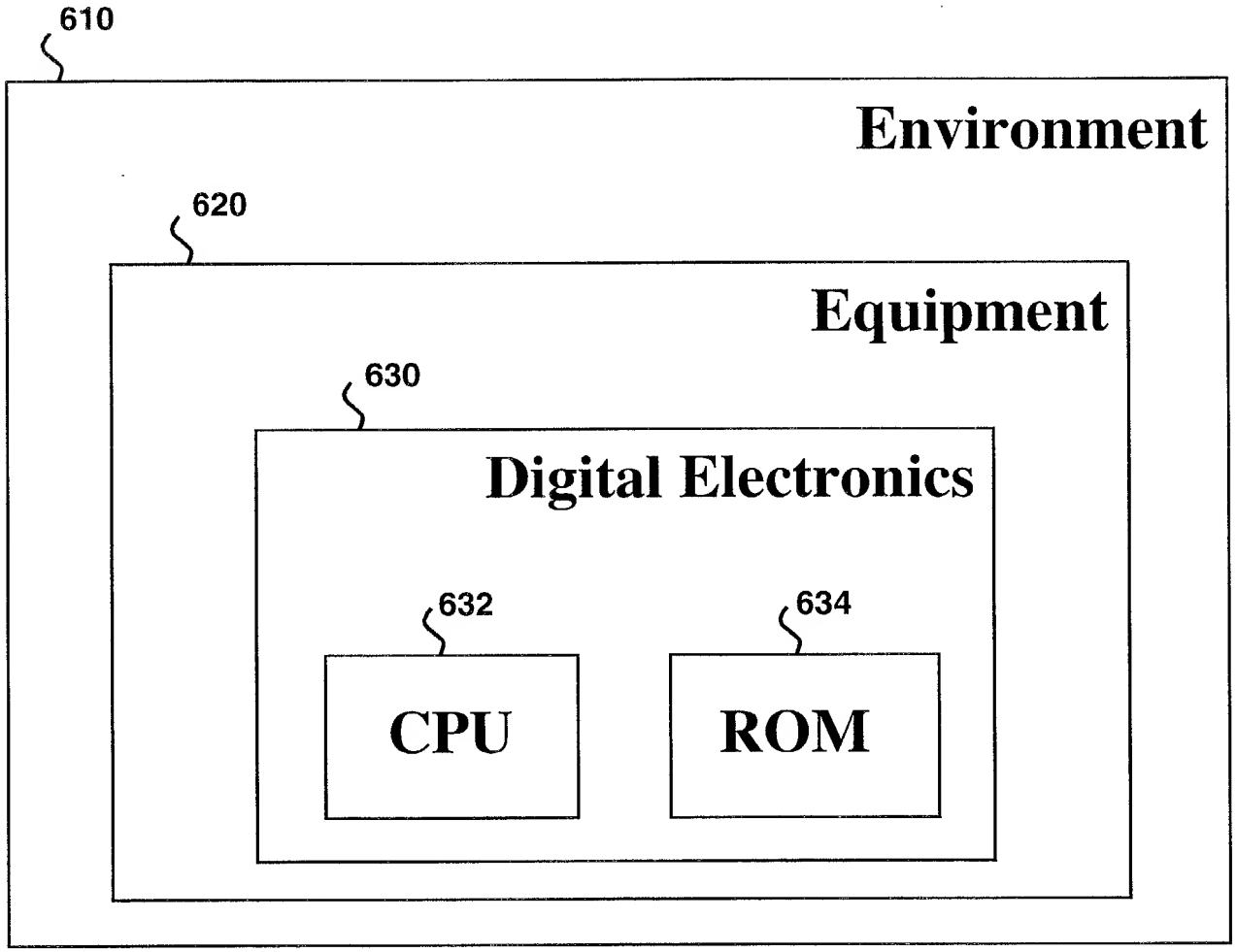

610
Environment
620
Equipment
630
Digital Electronics
632
634
CPU
ROM

FIG. 6

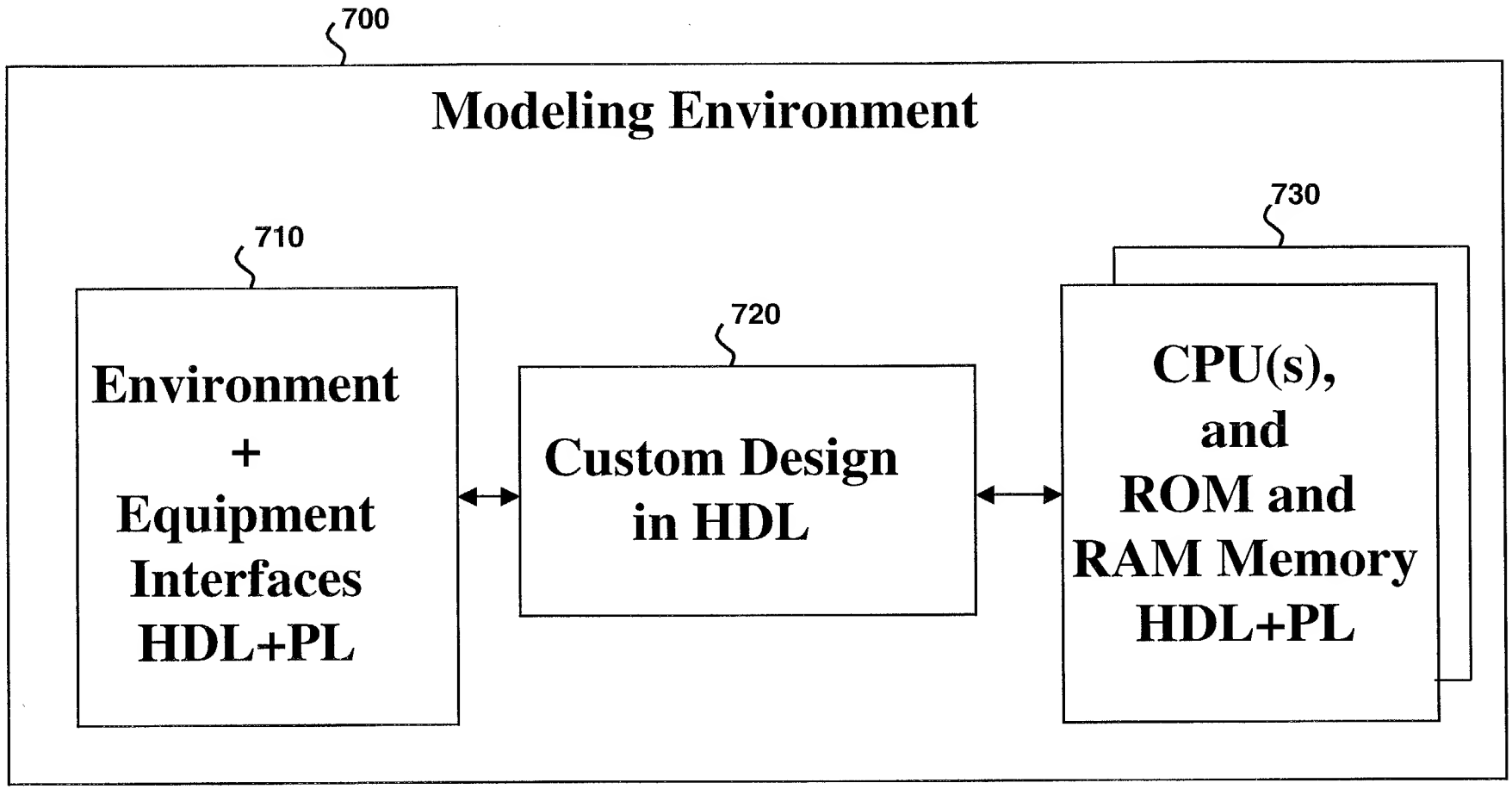

700
Modeling Environment
710
Environment
+
Equipment
Interfaces
HDL+PL
720
Custom Design
in HDL
730
CPU(s),
and
ROM and
RAM Memory
HDL+PL

FIG. 7

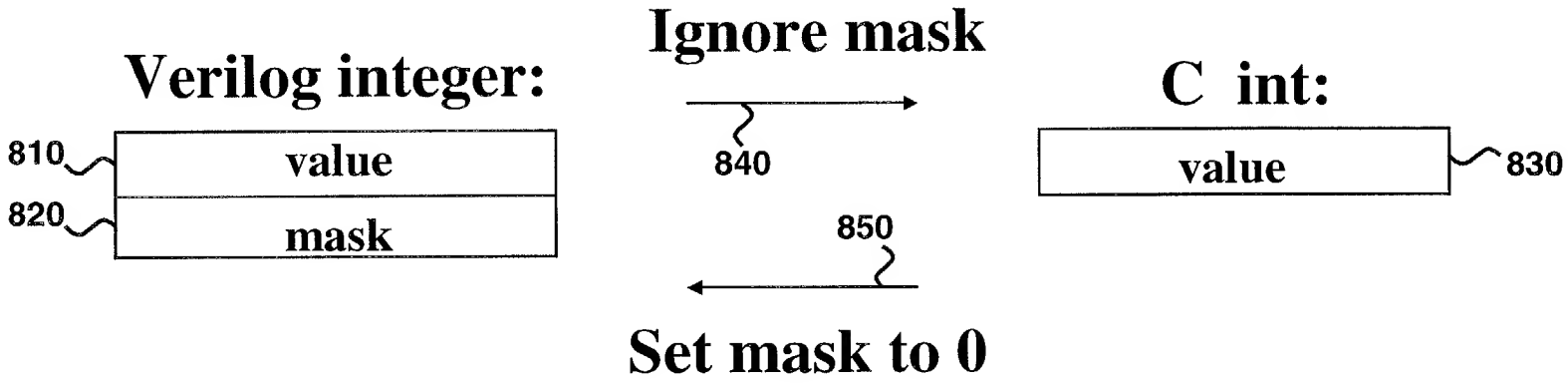
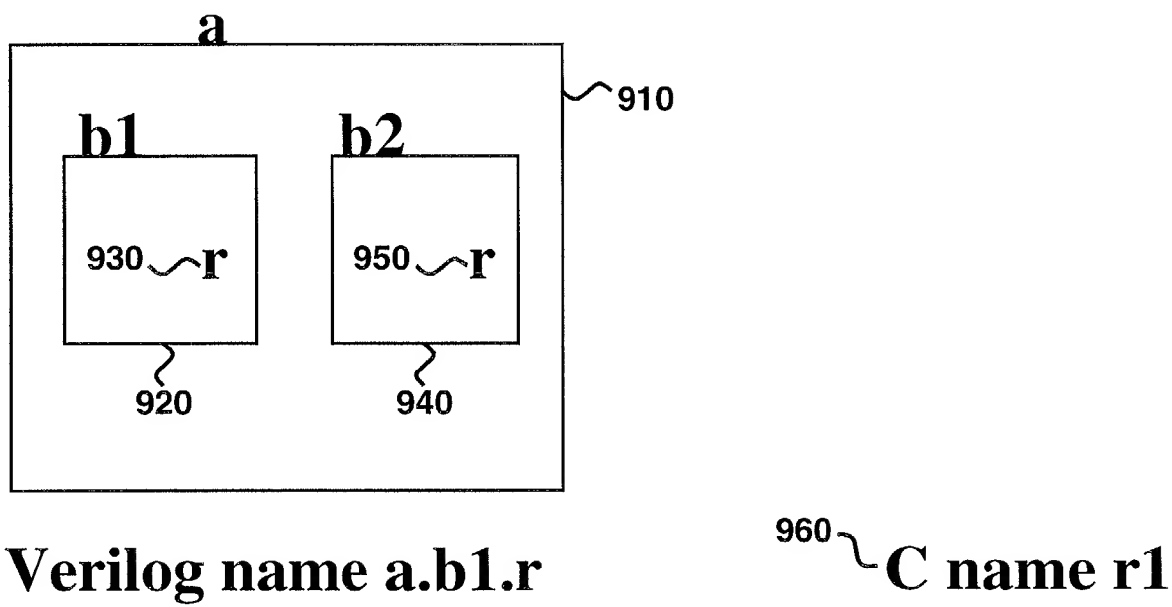

Verilog integer:
810
value
820
mask
Ignore mask
840
850
Set mask to 0
C int:
value
830

FIG. 8



Verilog name a.b1.r

960 C name r1

FIG. 9

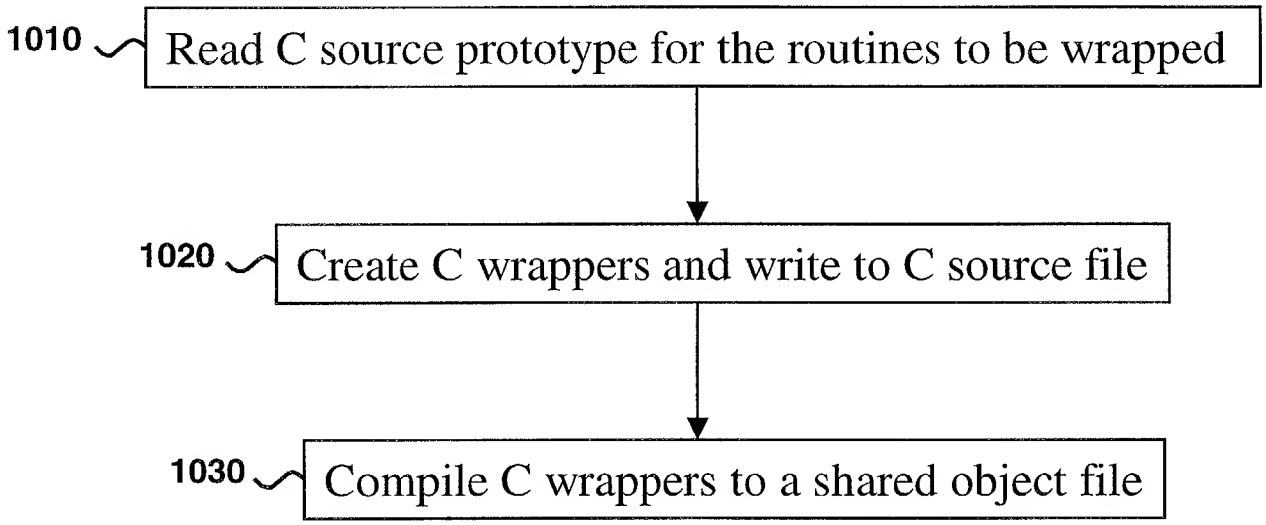

1010
Read C source prototype for the routines to be wrapped
1020
Create C wrappers and write to C source file
1030
Compile C wrappers to a shared object file

FIG. 10

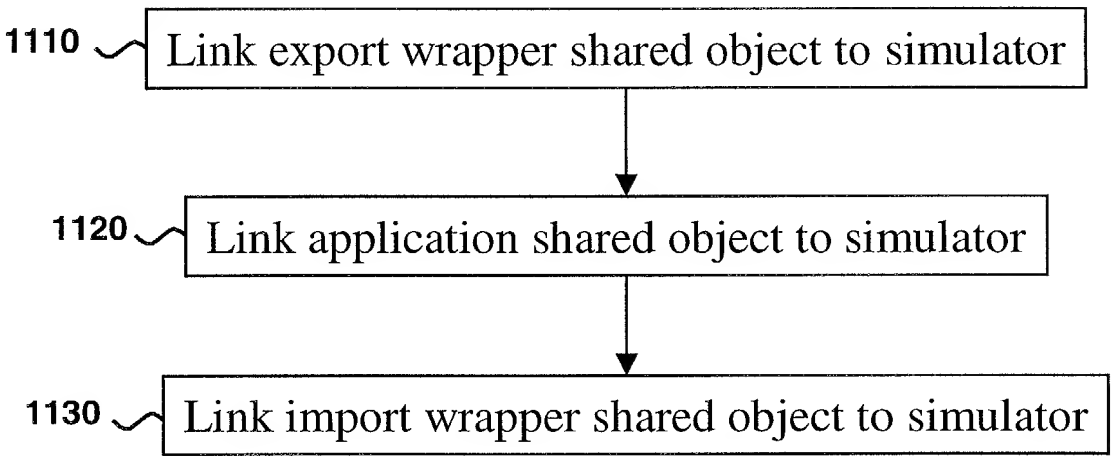

1110
Link export wrapper shared object to simulator
1120
Link application shared object to simulator
1130
Link import wrapper shared object to simulator

FIG. 11

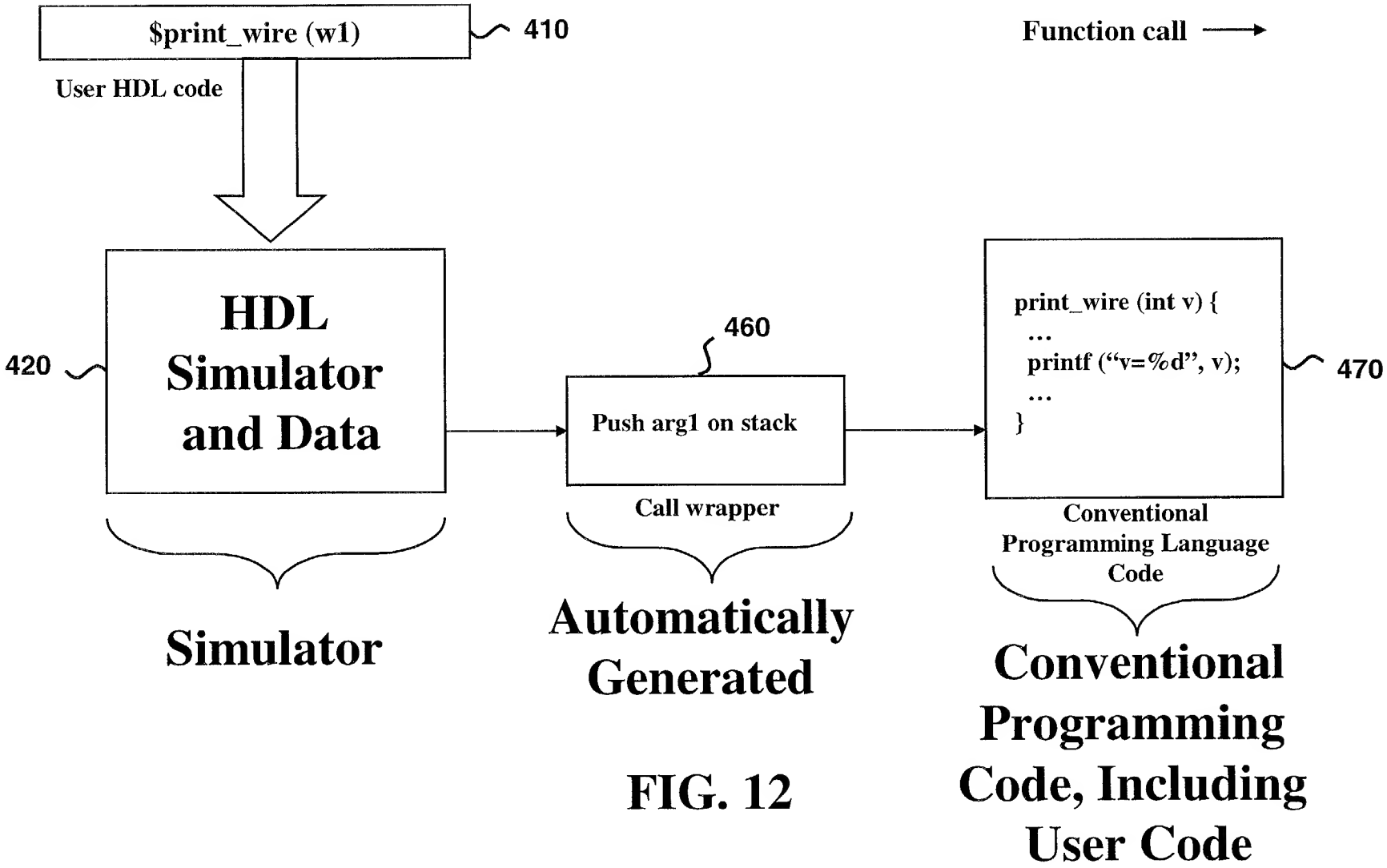

$print_wire (w1)
410
User HDL code
Function call
HDL Simulator and Data
420
460
Push arg1 on stack
Call wrapper
print_wire (int v) {
...
printf ("v=%d", v);
...
}
470
Conventional Programming Language Code
Simulator
Automatically Generated
Conventional Programming Code, Including User Code

FIG. 12

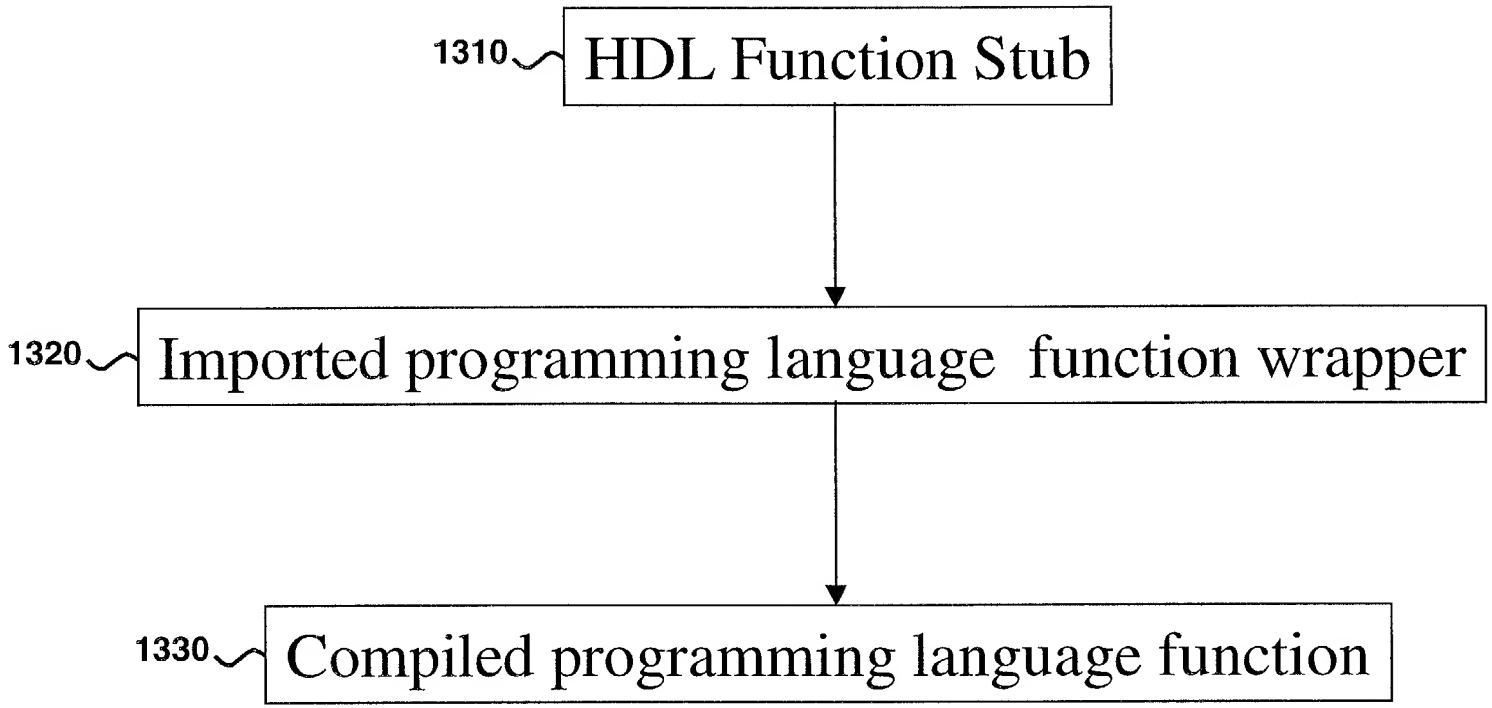

1310
HDL Function Stub
1320
Imported programming language function wrapper
1330
Compiled programming language function

FIG. 13

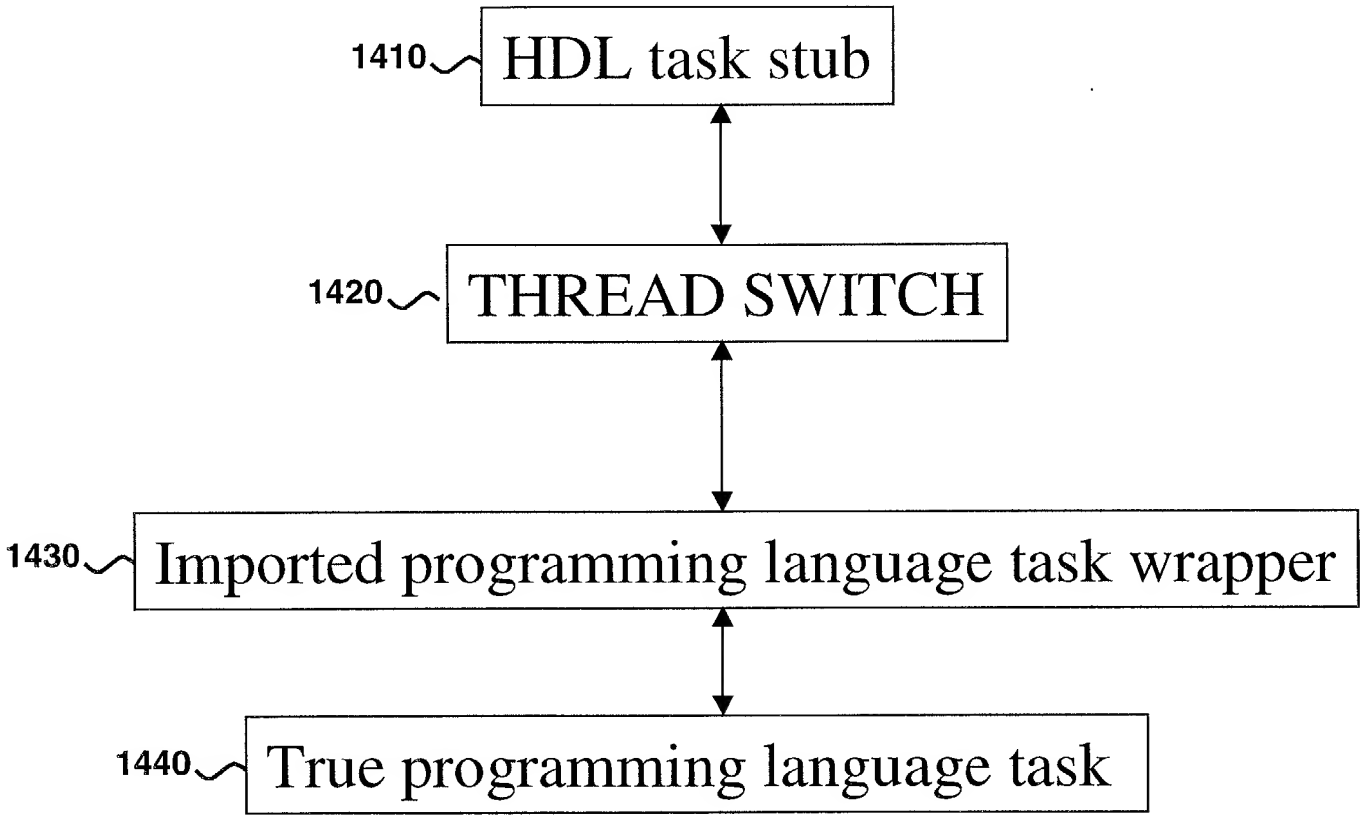

1410
HDL task stub
1420
THREAD SWITCH
1430
Imported programming language task wrapper
1440
True programming language task

FIG. 14

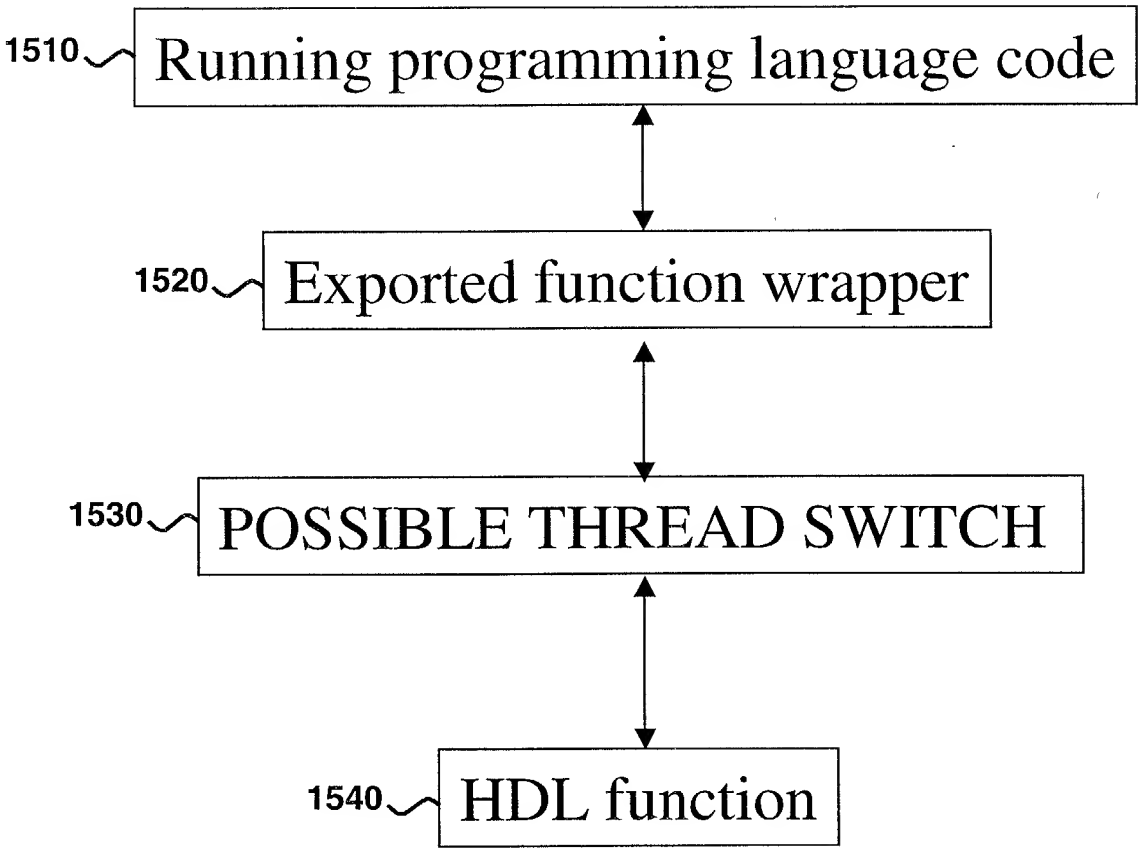

1510
Running programming language code
1520
Exported function wrapper
1530
POSSIBLE THREAD SWITCH
1540
HDL function

FIG. 15

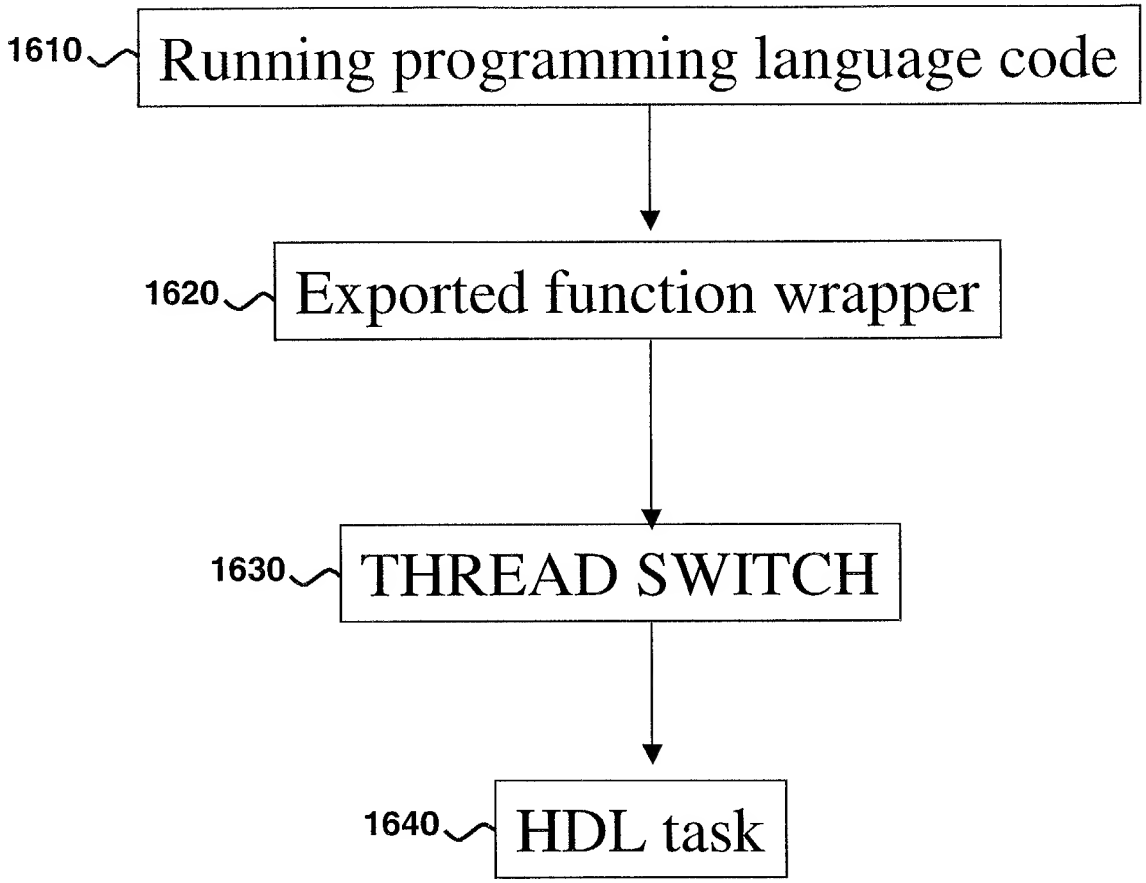

1610
Running programming language code
1620
Exported function wrapper
1630
THREAD SWITCH
1640
HDL task

FIG. 16

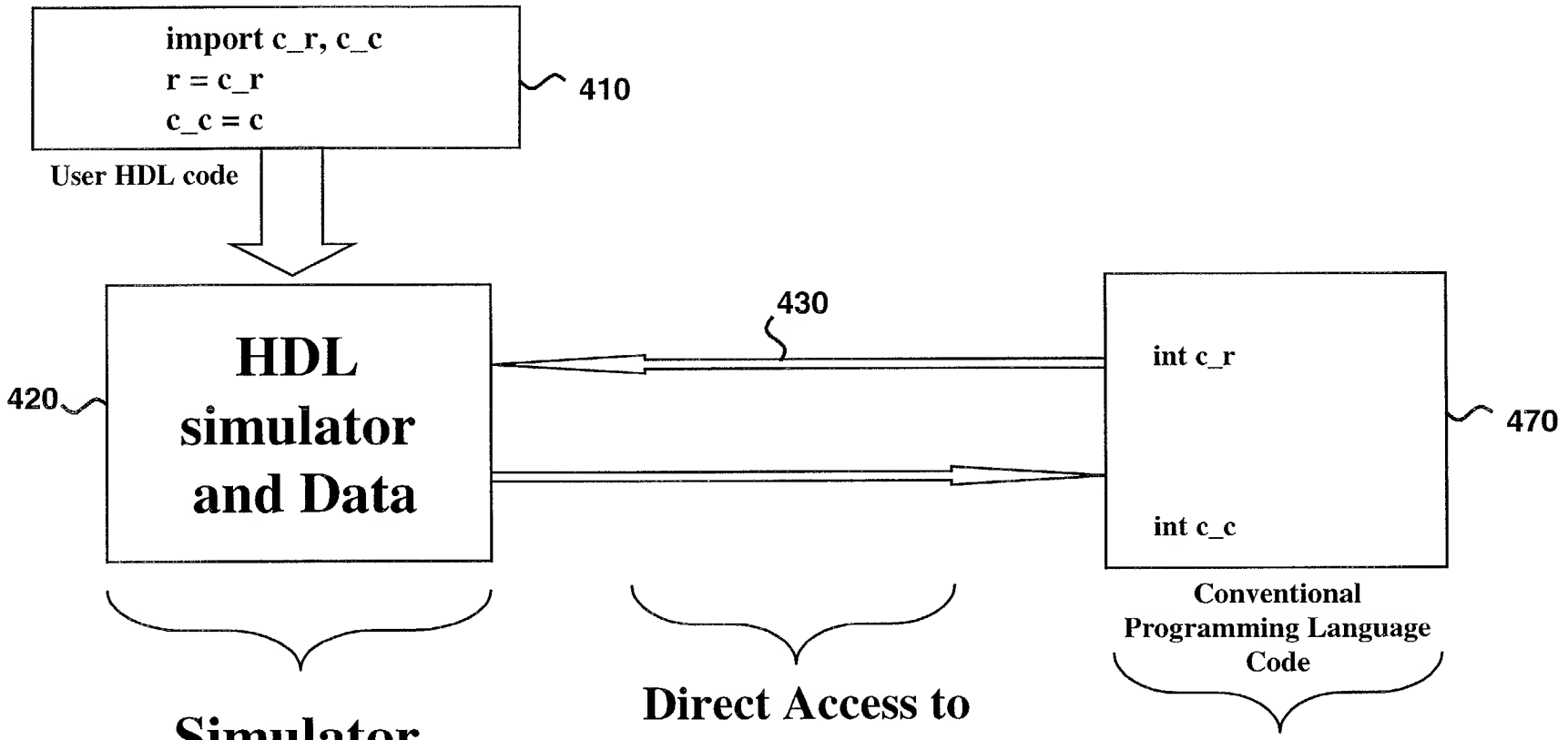

import c_r, c_c
r = c_r
c_c = c
410
User HDL code
HDL simulator and Data
420
430
int c_r
int c_c
470
Conventional Programming Language Code
Simulator
Direct Access to Programming Language Variables
Conventional Programming Code, Including User Code

FIG. 17

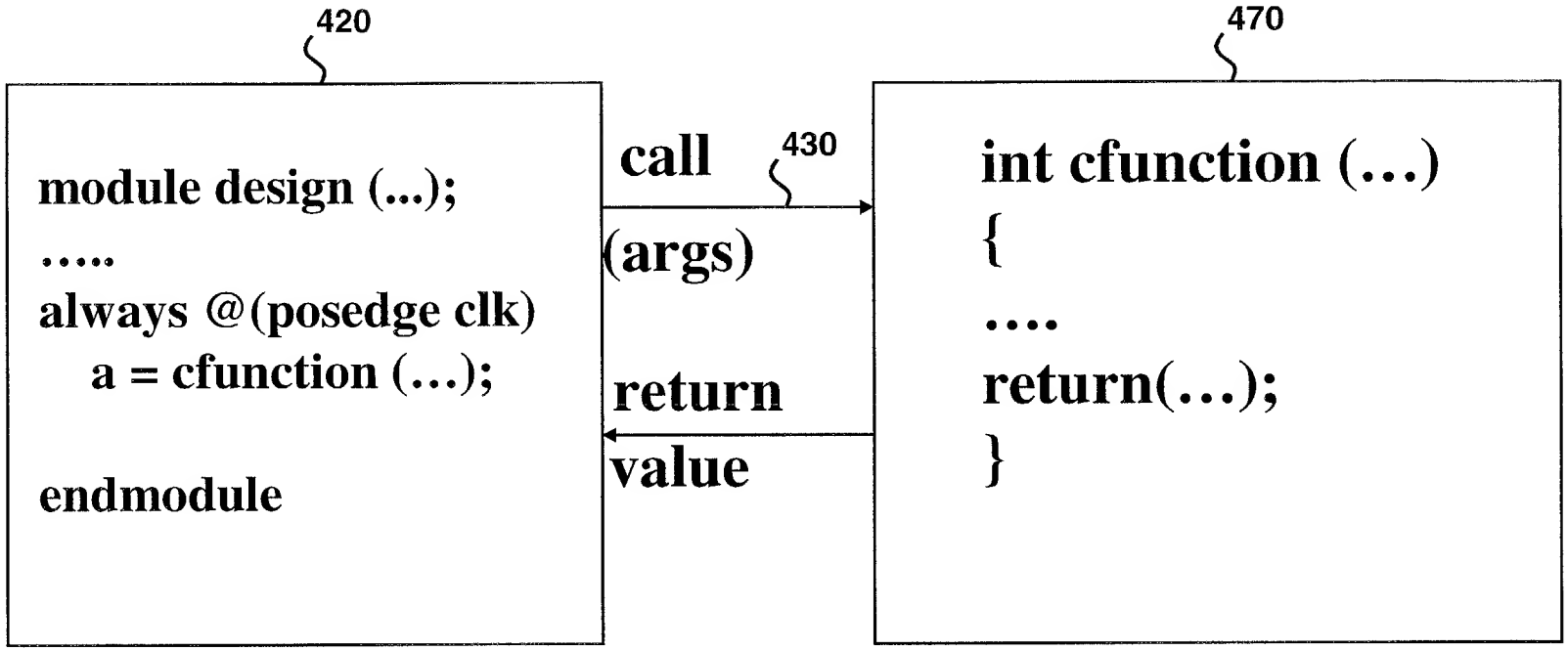

420
470
module design (...);
.....
always @(posedge clk)
a = cfunction (...);
endmodule
call
430
(args)
return
value
int cfunction (...)
{
....
return(...);
}

FIG. 18

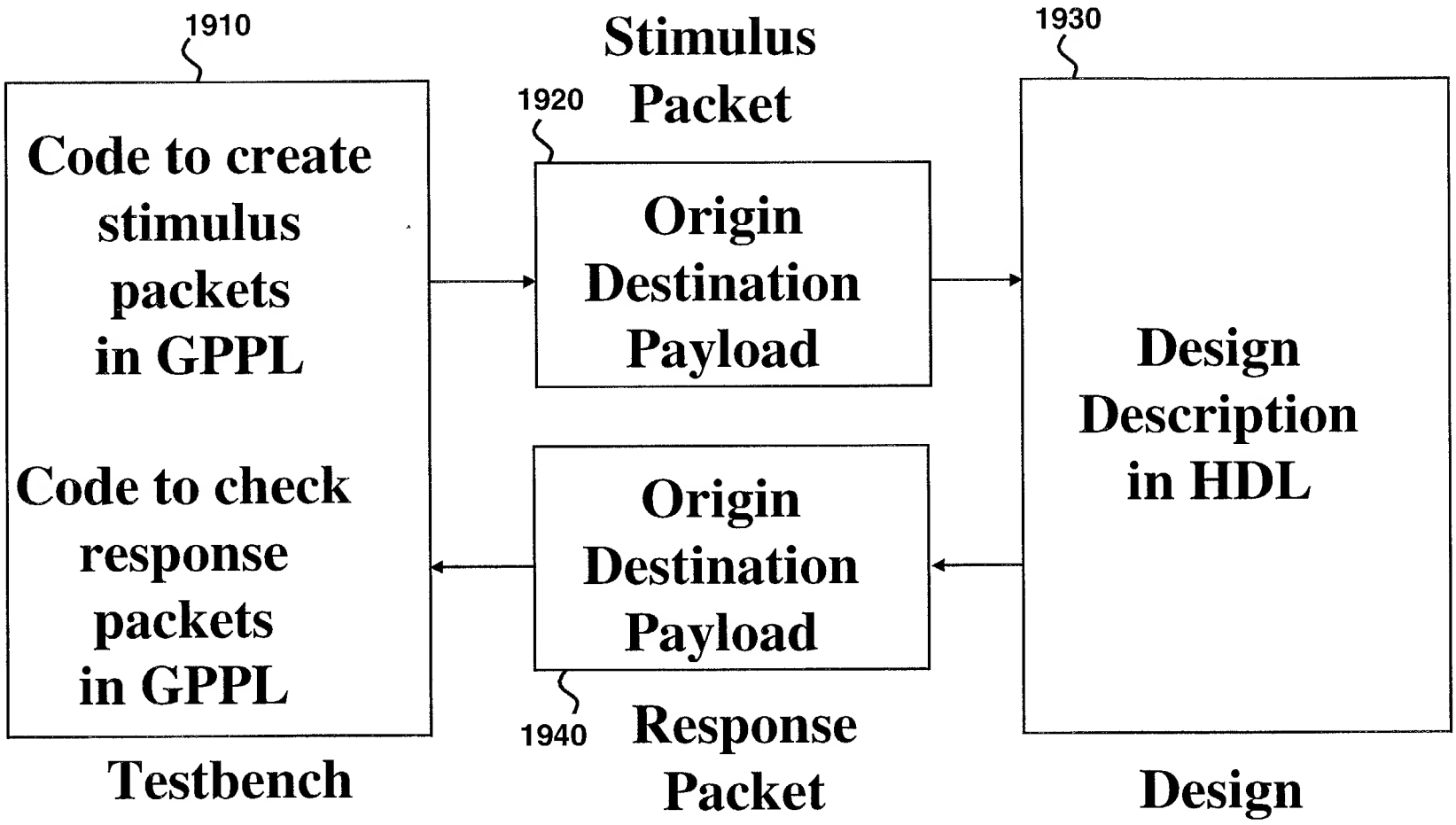

1910
Code to create stimulus packets in GPPL
Code to check response packets in GPPL
Testbench
1920
Stimulus Packet
Origin Destination Payload
Origin Destination Payload
1940
Response Packet
1930
Design Description in HDL
Design

FIG. 19